Guide complet : Déploiement Django sur 2 VMs Debian

Phase 1 : Développement sous Windows

1.1 Installation des outils sur Windows

1. **Installer Python** (3.8+)
   - Télécharger depuis python.org
   - Cocher "Add Python to PATH"

2. **Installer PostgreSQL**
   - Télécharger PostgreSQL depuis postgresql.org
   - Noter le mot de passe du superutilisateur

3. **Installer Git** (pour la gestion de version)

1.2 Création du projet Django

```bash
# Créer un environnement virtuel
python -m venv venv
venv\Scripts\activate

# Installer Django et les dépendances
pip install django psycopg2-binary python-decouple pillow

# Créer le projet
django-admin startproject gestion_absences
cd gestion_absences

# Créer une app pour la gestion
python manage.py startapp absences
```

1.3 Configuration de la base de données

1. **Créer la base de données PostgreSQL**
   ```sql
   -- Se connecter à PostgreSQL (pgAdmin ou psql)
   CREATE DATABASE gestion_absences_db;
   CREATE USER gestion_user WITH PASSWORD 'motdepasse123';
   GRANT ALL PRIVILEGES ON DATABASE gestion_absences_db TO gestion_user;
   ```

2. **Configurer Django (settings.py)**
   ```python
   from decouple import config
   import os

   INSTALLED_APPS = [
       'django.contrib.admin',
       'django.contrib.auth',
       'django.contrib.contenttypes',
       'django.contrib.sessions',
       'django.contrib.messages',
       'django.contrib.staticfiles',
       'absences',  # Notre app
   ]

   DATABASES = {
   ```

```
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': config('DB_NAME', default='gestion_absences_db'),
        'USER': config('DB_USER', default='gestion_user'),
        'PASSWORD': config('DB_PASSWORD', default='motdepasse123'),
        'HOST': config('DB_HOST', default='localhost'),
        'PORT': config('DB_PORT', default='5432'),
    }
}

ALLOWED_HOSTS = ['localhost', '127.0.0.1', 'IP_VM_DJANGO']

# Configuration pour les fichiers média (photos)
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

3. **Créer un fichier .env**
```
DB_NAME=gestion_absences_db
DB_USER=gestion_user
DB_PASSWORD=motdepasse123
DB_HOST=localhost
DB_PORT=5432
SECRET_KEY=votre_secret_key_django
DEBUG=True
```

4. **Créer les modèles Django (absences/models.py)**
```python
from django.db import models
from django.core.validators import EmailValidator

class GroupeEtudiant(models.Model):
    nom = models.CharField(max_length=100, unique=True)

    def __str__(self):
        return self.nom

    class Meta:
        verbose_name = "Groupe d'étudiants"
        verbose_name_plural = "Groupes d'étudiants"

class Etudiant(models.Model):
    nom = models.CharField(max_length=100)
    prenom = models.CharField(max_length=100)
    email = models.EmailField(unique=True, validators=[EmailValidator()])
    groupe = models.ForeignKey(GroupeEtudiant, on_delete=models.CASCADE,
related_name='etudiants')
    photo = models.ImageField(upload_to='photos/etudiants/', blank=True, null=True)

    def __str__(self):
        return f"{self.prenom} {self.nom}"

    class Meta:
        verbose_name = "Étudiant"
        verbose_name_plural = "Étudiants"
        unique_together = ['nom', 'prenom', 'groupe']  # Éviter les doublons

class Enseignant(models.Model):
```

```
    nom = models.CharField(max_length=100)
    prenom = models.CharField(max_length=100)
    email = models.EmailField(unique=True, validators=[EmailValidator()])

    def __str__(self):
        return f"{self.prenom} {self.nom}"

    class Meta:
        verbose_name = "Enseignant"
        verbose_name_plural = "Enseignants"

class Cours(models.Model):
    titre = models.CharField(max_length=200)
    date = models.DateTimeField()
    enseignant = models.ForeignKey(Enseignant, on_delete=models.CASCADE,
related_name='cours')
    duree = models.DurationField(help_text="Format: HH:MM:SS")
    groupe = models.ForeignKey(GroupeEtudiant, on_delete=models.CASCADE,
related_name='cours')

    def __str__(self):
        return f"{self.titre} - {self.date.strftime('%d/%m/%Y %H:%M')}"

    class Meta:
        verbose_name = "Cours"
        verbose_name_plural = "Cours"
        ordering = ['-date']

class Absence(models.Model):
    JUSTIFICATION_CHOICES = [
        ('justifie', 'Justifié'),
        ('non_justifie', 'Non justifié'),
        ('en_attente', 'En attente de justification'),
    ]

    etudiant = models.ForeignKey(Etudiant, on_delete=models.CASCADE,
related_name='absences')
    cours = models.ForeignKey(Cours, on_delete=models.CASCADE, related_name='absences')
    statut = models.CharField(max_length=20, choices=JUSTIFICATION_CHOICES,
default='non_justifie')
    justification = models.TextField(blank=True, null=True, help_text="Motif de l'absence")
    date_creation = models.DateTimeField(auto_now_add=True)
    date_modification = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"{self.etudiant} - {self.cours.titre} ({self.get_statut_display()})"

    class Meta:
        verbose_name = "Absence"
        verbose_name_plural = "Absences"
        unique_together = ['etudiant', 'cours']  # Un étudiant ne peut avoir qu'une absence par
cours
        ordering = ['-date_creation']
```

1.4 Développement et test

```bash
# Créer les migrations
python manage.py makemigrations absences
```

```
python manage.py migrate

# Créer un superutilisateur
python manage.py createsuperuser

# Tester le serveur
python manage.py runserver
```

Configurer l'administration Django (absences/admin.py)
```python
from django.contrib import admin
from .models import GroupeEtudiant, Etudiant, Enseignant, Cours, Absence

@admin.register(GroupeEtudiant)
class GroupeEtudiantAdmin(admin.ModelAdmin):
    list_display = ['nom']
    search_fields = ['nom']

@admin.register(Etudiant)
class EtudiantAdmin(admin.ModelAdmin):
    list_display = ['nom', 'prenom', 'email', 'groupe']
    list_filter = ['groupe']
    search_fields = ['nom', 'prenom', 'email']
    list_per_page = 20

@admin.register(Enseignant)
class EnseignantAdmin(admin.ModelAdmin):
    list_display = ['nom', 'prenom', 'email']
    search_fields = ['nom', 'prenom', 'email']

@admin.register(Cours)
class CoursAdmin(admin.ModelAdmin):
    list_display = ['titre', 'date', 'enseignant', 'groupe', 'duree']
    list_filter = ['enseignant', 'groupe', 'date']
    search_fields = ['titre']
    date_hierarchy = 'date'

@admin.register(Absence)
class AbsenceAdmin(admin.ModelAdmin):
    list_display = ['etudiant', 'cours', 'statut', 'date_creation']
    list_filter = ['statut', 'cours__groupe', 'date_creation']
    search_fields = ['etudiant__nom', 'etudiant__prenom', 'cours__titre']
    readonly_fields = ['date_creation', 'date_modification']
```

Créer des données de test (absences/management/commands/create_test_data.py)
```python
# Créer le dossier: absences/management/commands/
from django.core.management.base import BaseCommand
from absences.models import GroupeEtudiant, Etudiant, Enseignant, Cours, Absence
from datetime import datetime, timedelta
from django.utils import timezone

class Command(BaseCommand):
    help = 'Créer des données de test'

    def handle(self, *args, **options):
        # Créer des groupes
        groupe1 = GroupeEtudiant.objects.create(nom="INFO-L3-A")
```

```
        groupe2 = GroupeEtudiant.objects.create(nom="INFO-L3-B")

        # Créer des étudiants
        Etudiant.objects.create(nom="Dupont", prenom="Jean", email="jean.dupont@email.com",
groupe=groupe1)
        Etudiant.objects.create(nom="Martin", prenom="Marie", email="marie.martin@email.com",
groupe=groupe1)
        Etudiant.objects.create(nom="Durand", prenom="Pierre", email="pierre.durand@email.com",
groupe=groupe2)

        # Créer des enseignants
        prof1 = Enseignant.objects.create(nom="Professeur", prenom="Paul",
email="paul.prof@univ.fr")
        prof2 = Enseignant.objects.create(nom="Docteur", prenom="Anne",
email="anne.doc@univ.fr")

        # Créer des cours
        cours1 = Cours.objects.create(
            titre="Programmation Web",
            date=timezone.now(),
            enseignant=prof1,
            duree=timedelta(hours=2),
            groupe=groupe1
        )

        self.stdout.write(self.style.SUCCESS('Données de test créées avec succès'))
```

Lancer la création de données test:
```bash
python manage.py create_test_data
```

1.5 Préparer les fichiers pour le déploiement

```bash
# Générer requirements.txt
pip freeze > requirements.txt

# Créer un script de démarrage
# start_server.sh
#!/bin/bash
source venv/bin/activate
python manage.py migrate
python manage.py collectstatic --noinput
python manage.py runserver 0.0.0.0:8000
```

Phase 2 : Préparation des VMs Debian

2.1 Configuration VM 1 (Base de données)

```bash
# Mise à jour du système
sudo apt update && sudo apt upgrade -y

# Installation PostgreSQL
sudo apt install postgresql postgresql-contrib -y

# Configuration PostgreSQL
```

```
sudo systemctl start postgresql
sudo systemctl enable postgresql

# Configuration utilisateur PostgreSQL
sudo -u postgres psql
```

```sql
-- Dans psql
CREATE DATABASE gestion_absences_db;
CREATE USER gestion_user WITH PASSWORD 'motdepasse123';
GRANT ALL PRIVILEGES ON DATABASE gestion_absences_db TO gestion_user;
\q
```

Configuration réseau PostgreSQL :
```bash
# Éditer postgresql.conf
sudo nano /etc/postgresql/*/main/postgresql.conf
# Modifier : listen_addresses = '*'

# Éditer pg_hba.conf
sudo nano /etc/postgresql/*/main/pg_hba.conf
# Ajouter : host all all IP_VM_DJANGO/32 md5

# Redémarrer PostgreSQL
sudo systemctl restart postgresql

# Ouvrir le port 5432
sudo ufw allow 5432
```

2.2 Configuration VM 2 (Application Django)

```bash
# Mise à jour du système
sudo apt update && sudo apt upgrade -y

# Installation des paquets nécessaires
sudo apt install python3 python3-pip python3-venv git nginx -y

# Création du répertoire projet
sudo mkdir /var/www/monprojet
sudo chown $USER:$USER /var/www/monprojet
cd /var/www/monprojet
```

Phase 3 : Transfert et déploiement

3.1 Transférer les fichiers

Option 1 : SCP (depuis Windows)
```bash
# Depuis PowerShell/CMD Windows
scp -r C:\chemin\vers\gestion_absences user@IP_VM_DJANGO:/var/www/
```

Option 2 : Git (recommandé)
```bash
# Sur la VM Django
```

```
cd /var/www/gestion_absences
git clone https://github.com/votre-repo/gestion_absences.git .
```

3.2 Configuration sur VM Django

```bash
# Créer l'environnement virtuel
python3 -m venv venv
source venv/bin/activate

# Installer les dépendances
pip install -r requirements.txt

# Modifier le fichier .env
nano .env
```

```
DB_NAME=gestion_absences_db
DB_USER=gestion_user
DB_PASSWORD=motdepasse123
DB_HOST=IP_VM_DATABASE
DB_PORT=5432
SECRET_KEY=votre_secret_key_django
DEBUG=False
```

3.3 Configuration Nginx

```bash
# Créer la configuration Nginx
sudo nano /etc/nginx/sites-available/gestion_absences
```

```nginx
server {
    listen 80;
    server_name IP_VM_DJANGO;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /static/ {
        alias /var/www/gestion_absences/static/;
    }

    location /media/ {
        alias /var/www/gestion_absences/media/;
    }
}
```

```bash
# Activer le site
sudo ln -s /etc/nginx/sites-available/gestion_absences /etc/nginx/sites-enabled/
```

```
sudo nginx -t
sudo systemctl restart nginx
```

3.4 Configuration Gunicorn (recommandé pour production)

```bash
# Installer Gunicorn
pip install gunicorn

# Créer un fichier de service systemd
sudo nano /etc/systemd/system/monprojet.service
```

```ini
[Unit]
Description=Gunicorn instance to serve monprojet
After=network.target

[Service]
User=www-data
Group=www-data
WorkingDirectory=/var/www/monprojet
Environment="PATH=/var/www/monprojet/venv/bin"
ExecStart=/var/www/monprojet/venv/bin/gunicorn --workers 3 --bind 127.0.0.1:8000
monprojet.wsgi:application
Restart=always

[Install]
WantedBy=multi-user.target
```

```bash
# Démarrer le service
sudo systemctl daemon-reload
sudo systemctl start monprojet
sudo systemctl enable monprojet
```

Phase 4 : Tests et finalisation

4.1 Tests de connectivité

```bash
# Sur VM Django, tester la connexion BDD
python manage.py dbshell

# Vérifier les migrations
python manage.py migrate

# Collecter les fichiers statiques
python manage.py collectstatic
```

4.2 Configuration du pare-feu

VM Database :
```bash
sudo ufw allow 22      # SSH
sudo ufw allow 5432    # PostgreSQL
```

sudo ufw enable
```

**VM Django :**
```bash
sudo ufw allow 22      # SSH
sudo ufw allow 80      # HTTP
sudo ufw allow 443     # HTTPS (si certificat SSL)
sudo ufw enable
```

4.3 Sauvegarde et monitoring

```bash
# Script de sauvegarde BDD
#!/bin/bash
pg_dump -h IP_VM_DATABASE -U gestion_user gestion_absences_db > backup_$(date +
%Y%m%d_%H%M%S).sql

# Ajouter au crontab pour sauvegarde automatique
crontab -e
# 0 2 * * * /chemin/vers/backup_script.sh
```

Résumé des adresses IP à configurer

- `IP_VM_DATABASE` : Adresse IP de la VM base de données
- `IP_VM_DJANGO` : Adresse IP de la VM application Django
- Modifier tous les fichiers de configuration avec ces IPs

Commandes de dépannage

```bash
# Vérifier les logs Django
sudo journalctl -u gestion_absences -f

# Vérifier les logs Nginx
sudo tail -f /var/log/nginx/error.log

# Vérifier les logs PostgreSQL
sudo tail -f /var/log/postgresql/postgresql-*.log

# Tester la connexion BDD depuis VM Django
telnet IP_VM_DATABASE 5432
```

Phase 5 : Fonctionnalités avancées spécifiques

5.1 Créer des vues personnalisées (absences/views.py)

```python
from django.shortcuts import render, get_object_or_404
from django.http import JsonResponse
from django.views.generic import ListView
from .models import Etudiant, Cours, Absence, GroupeEtudiant

class AbsencesByGroupeView(ListView):
    model = Absence
    template_name = 'absences/absences_by_groupe.html'
    context_object_name = 'absences'
```
```

```
    def get_queryset(self):
        groupe_id = self.kwargs.get('groupe_id')
        return Absence.objects.filter(cours__groupe_id=groupe_id).select_related('etudiant', 'cours')

def statistiques_absences(request):
    stats = {
        'total_absences': Absence.objects.count(),
        'absences_justifiees': Absence.objects.filter(statut='justifie').count(),
        'absences_non_justifiees': Absence.objects.filter(statut='non_justifie').count(),
        'en_attente': Absence.objects.filter(statut='en_attente').count(),
    }
    return JsonResponse(stats)
```

5.2 Créer des URLs (absences/urls.py)

```python
from django.urls import path
from . import views

app_name = 'absences'

urlpatterns = [
    path('groupe/<int:groupe_id>/absences/', views.AbsencesByGroupeView.as_view(),
name='absences_by_groupe'),
    path('api/stats/', views.statistiques_absences, name='stats_absences'),
]
```

5.3 Configuration des URLs principales (gestion_absences/urls.py)

```python
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('absences/', include('absences.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```